REMARKS

In the Office Action, the Examiner rejected claim 5 under section 101, rejected claims 1 and 6 under the first paragraph of section 112, rejected claim 5 as anticipated by the Prashant et al. reference, rejected claim 1 as obvious over Prashant et al in view of Monday et al., rejected claim 2 as obvious over Prashant et al in view of Monday and further in view of Ford et al., rejected claim 3 as obvious over Prashant in view of Monday and further in view of Gamma et al., rejected claim 4 as obvious over Prashant in view of Monday and further in view of Schmidt, rejected claim 6 as obvious over Prashant in view of Monday and Goldsmith et al in further view of Northrup et al.

35 USC §101

Applicants have amended the claims to address the section 101 issue raised by the Examiner. In particular, the claim 5 as amended claims inserting an generic main component, thereby providing a tangible aspect to the claimed invention. A tangible result is provided as well, wherein the programs on the computer were unconfigured prior to the method being performed and after the method the services are configured. Those programs now host the generic main component and thus the configured services. The hosting program functionality has been enriched by configurable services.

35 USC §112, 1st ¶

The phrase "business logic layer" is a common term of art for those in the software field, and would have been well known to a man skilled in the art at the time of filing the present application. Attached are two examples of the term being used in the December 1998 publication Microsoft Systems Journal Editor's Note at http://www.microsoft.com/msj/1298/ednote/ednote1298.aspx (see third paragraph, line 6) and Byte.com article "Enterprise Visual Basic? Almost..."of October 1996 at http://www.byte.com/art/9610/sec12/art8.htm (see third paragraph). A copy of each of these two articles is enclosed as Appendix A for the Examiner's review. The phrase is still in use as shown by the web site http://en.wikipedia.org/wiki/Business_logic_layer.

Through the rephrase to "business logic layer component" this part will be clear.

35 USC §102

The publication of Prashant Jain and Douglas C. Schmidt (referred to as Prashant by the Examiner) discloses a dynamically configured service configurator pattern. The reference presents the structure of the services configurator pattern, providing samples in the time server class and the clerk class. The reference does not, however, teach componentization of the Service Configurator into a generic main object (= compiled version of the GenericMain class).

The disclosed code (class GenericMain) allows to be packaged into a DLL or ActiveX control, such that a host program can be enriched with this functionality (this is not explicitly claimed, but inherently assumed throughout the text).

Contrary to what is taught in each of the prior art references cited in the action, the present application does not related to communications. As such, the present invention differs from the cited art.

The present application does not relate to a connector between two frameworks, but instead relates to a framework interaction. For example, see Fig. 5 of the drawings which shows an MFC-ACE Framework Interaction. Claims 1 and 6 claim a framework connector.

In the present application, for example, in the upper part of Fig. 5 is seen UI-Logic (user interface logic, or presentation logic) and in the lower part Business-Logic. Presentation logic and business logic is claimed in claims 1 and 6. Prashant et al. (Jain et al.) does not teach UI-Logic and Business-Logic. As such, the invention is not anticipated nor obvious over the cited reference.

The Prashant et al. (Jain et al.) reference teaches to establish and maintain connections with one or more time servers (see page 8, left column, first paragraph, where connections are made to the clerk component in a distributed time service). The Prashant (Jain) reference does not teach "a framework connector providing communications between components", contrary to the assertions of the Examiner. See the language of claim 1 which call for this feature and thus distinguishes over the cited reference.

As to claim 5, Prashant et al. does not teach a generic main as an framework connector, as provided in the present application and as shown in Figs. 5 and 7 (see point 4 of Amendment A).

35 USC §103

By no means does the reference to Monday et al. teach threading and collaboration issues of various frameworks as shown in Figure 5 of the present application. The framework connector ensures the thread affinity of interacting components (e.g. ActiveX components must always be invoked in the thread context where it has been created).

By contrast, the Monday reference describes a system requiring a visual construction simplification mechanism (as in claim 1) which is by no means similar to the framework connector as described in the present application.

Regarding the rejection over the combination including Schmidt, the framework connector provides thread routing and synchronization too.

Regarding the rejection over the combination including Goldsmith and Northrup, Applicants respectfully submit that page 10, second to forth paragraphs, of the office action is incorrect. None of the references disclose a "message pump interconnection protocol" according to claim 6.

As to the prior art in general,

- The framework connector, which the examiner indicates is found in the reference to Monday, has no similarity to the framework connector according the present patent application. As Claim 6 points out, the generic connection mechanism with GUI frameworks provides the runtime environment (main()) (over the message pump interaction protocol). The reference to Monday does not have such a feature.
- Furthermore, Prashant did not describe, how for example the ActiveX Controls in a container process is to be loaded via ACE_Service_Configurator and how their lifecycle is controlled (all features of the framework connectors).

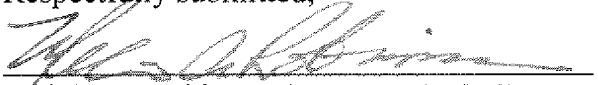
In the present application, the framework connector is claimed in claim1. The arguments of the examiner do not apply. No prior art (or a combination thereof) describes a system according the present application.

The cited references, whether considered alone or in combination, do not show or suggest the invention and disclosed and claimed in the present application. As such, the invention is non-obvious over the art.

Conclusion

Applicants respectfully request reconsideration and allowance of the present application in view of the foregoing.

Respectfully submitted,



Melvin A. Robinson (Reg. No. 31,870)
Schiff Hardin LLP
Patent Department
6600 Sears Tower
Chicago, Illinois 60606
Telephone: 312-258-5785
CUSTOMER NO. 26574
ATTORNEY FOR APPLICANT

CH1\ 5035077.1

APPENDIX A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT:	Karlheinz DORN et al.	GROUP ART UNIT: 2194
SERIAL NO.:	09/215,732	EXAMINER: Charles E. Anya
FILING DATE:	December 18, 1998	CONFIRMATION NO.: 1714
INVENTION:	"INTERPRETIVE NETWORK DAEMON IMPLEMENTED BY GENERIC MAIN OBJECT"	

December 1998

MICROSOFT SYSTEMS JOURNAL

EDITOR'S NOTE

While this is the December issue that you're reading in November, it's actually early October and we're mile-high and dry in Denver at the Microsoft PDC. In addition to the cornucopia of Windows NT 5.0 material being presented, a major part of the PDC messaging this year involves the Windows DNA Architecture. That's Distributed interNet Application for the acronym-impaired.

Maybe you recall that we discussed Windows DNA in the editor's note of December 1997. Has anything changed since then? For the most part we got it right, but it's more refined from the marketing hype of last year. Windows DNA is the official application development model for the Windows platform.

The model is three-tiered, consisting of the Presentation, Business Logic, and Data layers. The Presentation layer houses the Windows clients. These applications range from the DHTML-based thin client to the Win32-based rich client. Why not thick client? The Presentation layer simply provides the user interface to the Business Logic layer. Your clients may or may not pass through a firewall to get the Business Logic layer, but this is where your scalable server applications churn through your bread-and-butter code. The server applications will access the Data layer, which can contain your legacy information on Windows NT Servers, Unix machines (deftly covered in this issue's article by Mai-lan Tomsen), IBM mainframes, or whatever.

Windows DNA technologies include COM+ and its associated services. Building upon existing COM and MTS, COM+ services provide load balancing, events, queued components, object pooling, and more. Watch for COM+ coverage in *MSJ* from our very own Don Box and others. Windows NT 5.0 plans to offer services such as the Active Directory and security features such as the Public Key Infrastructure, Kerberos, and Domain Policy.

Windows DNA encompasses the entire software architecture spectrum. Microsoft provides all the OS platforms, technologies, and development tools. If you can't wait for more information, check out the whitepapers, FAQs, and other goodies at http://www.microsoft.com/business/products/webplatform/.

Also in this issue, both Matt Pietrek and Jeffrey Richter give their own perspective on the latest cool feature of the Visual Studio linker: /DELAYLOAD. This technology is one of those brilliant ideas that seems so simple. We wonder why it previously hadn't been in the language products. While we'll leave the details to the experts, basically delay loading allows an EXE to only load a DLL when code execution from that DLL is needed. Ingenious! Before delay loading existed,

all implicitly linked DLLs were loaded when the executable first started.

Also in the news is the announcement of an upcoming service pack for Visual Studio 6.0. Several redistributable runtime DLLs will have bugs fixed, including MFC42.DLL, MSVCRT.DLL, and MSVBVM60.DLL. For the eventual release, stay tuned to the Visual Studio Web site, http://msdn.microsoft.com/vstudio/default.asp. In the meantime, our advice is to hold off redistributing the bits until the service pack ships.

— J.F.

CMP

Archives
Columns
Features
Print Archives 1994-1998

Special
BYTE Digest
Michael Abrash's *Graphics Programming Black Book*
101 Perl Articles

About Us
How to Access BYTE.com
Write to BYTE.com
Advertise with BYTE.com

ARTICLES

Enterprise Visual Basic? Almost..

Siemens Nixdorf's ComUnity aims to turn Visual Basic 4.0 into a client/server environment but falls short.

Volker Weber and Hans-Jochen Schmitt

Although Microsoft markets the Enterprise Edition of Visual Basic 4.0 (VB 4) as an enterprise-level client/server tool, the product still lacks the high-level business-logic tools of such client/server mainstays as PowerBuilder.

Now, along comes Siemens Nixdorf, the German computer company, with a major effort to provide an overarching framework for client/server development that exploits VB 4's open programmable objects. Called ComUnity, the framework prescribes how to build distributed client/server applications. The related development environment is called the Open Technology (OT) Framework, which in its fir st incarnati on works with VB 4. Siemens Nixdorf plans to port it to other languages, including C++. Our examination of a beta version revealed an ambitious product that nonetheless falls well short of its goals.

Members of the ComUnity

ComUnity consists of a three-layer application model that comprises a Presentation Layer, a Business Logic Layer, and a Database Layer (see the figure). This model, like other three-tier architectures, offers advantages in scalability, usability, maintenance, and performance compared to monolithic applications. You can distribute the layers across a network, with the Presentation Layer on users' desktop computers, the Business Logic Layer on an applications server, and the Database Layer on a database server.

To implement ComUnity's architecture in VB, Siemens Nixdorf had to employ Microsoft's standard development tools, including VB 4 itself,

Microsoft Visual C++ 2.2, and OLE components. But b ecause these tools are best suited for small-scale software projects, Siemens Nixdorf added more tools to improve support for large projects.

Inside the OT Framework

Central to OT Framework's Database Layer is a relational data dictionary that serves as a central repository where all the tables, attributes, and relations are managed. From the data dictionary, OT Framework generates Open Database Connectivity (ODBC) data sources, migrates databases, generates SQL statements for data access, and even produces simple entry forms for the user interface.

Unfortunately, even though you can manage the data dictionary with a simple data-entry and maintenance tool, OT Framework provides no graphical representation of the data-dictionary information. It does, however, import data models produced with CASE tools, such as Microtool's CASE 4.0 and IBM's Mood.

The data dictionary becomes part of the run-time environment repository that's distributed with OT Framework applications. Because t he data dictionary contains information on the relationships between tables, applications can provide for relational integrity even if the underlying database has no such provisions.

The Presentation Layer, which establishes the user interface (UI), is built around VB 4. A forms generator uses the data-dictionary information to generate default VB 4.0 data-entry forms. You can customize these forms, but they also work with real data right out of the box.

OLE Automation acts as the glue between the Presentation Layer UI and the Business Layer. Perl scripts link code and single UI controls, but the program generates them automatically, so users don't have to deal directly with Perl. A cross-checker monitors the UI forms against any modifications in the data dictionary.

The forms are stored in the run-time repository, and you can use OT Framework tools to further customize the UI by working on the forms data. Users can use the Tailoring Tool (see the screen) to customize the UI: data-entry fields can be switched off, changed, or moved, even in the delivered version of an application. Developers who are skittish about giving users this much control over the UI can turn the tailoring tool off. Regardless of whether a site contains some highly individualized UIs, centralized program updates can merge with them seamlessly.

Putting It All Together

As the figure indicates, the glue between the UI and the database is the Business Logic Layer. OT Framework implements this layer in VB 4 code using the new VB 4 class concept. This pseudo-object-oriented feature allows you to define reusable classes and instances, but it doesn't support inheritance or polymorphism. OT Framework comes with predefined classes for UI handling and general-purpose business applications, such as order entry.

The application framework itself is a template-based VB data dictionary called Code Wizards. Regrettably, except for a run-time

debugger, there are no tools in the Business Logic Layer -- you're on your own as far as business processes are concerned. Thus, any higher-level knowledge must go from paper directly into VB code.

An Incomplete Toolkit

If you're not familiar with the VB environment, ODBC data sources, or tool-based software development environments in general, installing ComUnity can be a bit daunting. The beta version of the setup routine had some rough edges; the ODBC data sources for the data dictionary and the target applications are not added automatically. You must also add the VB Code Wizards and Form Wizards manually to the VB environment.

From a lone developer's point of view, OT Framework's major advantage is its vast library of reusable VB code for business applications software. The tool is rather loosely integrated with the VB environment. For small-scale projects, it does not go much beyond VB 4 or other add-ins and components that are on the market.

ComUnity also lacks an integrated version-control and projec t management tool. It relies on Microsoft Visual Source Safe, which comes bundled with the VB 4 Professional Edition, for version control. But Siemens Nixdorf says it's working to integrate Microsoft Project into a future version of ComUnity OT Framework.

Missing from this first implementation of the ComUnity architecture are integrated Upper-CASE tools, such as data modelers and object-oriented modeling tools. Sinking the whole business logic into pure VB code is asking for trouble, especially if users ask for changes in the application.

As long as you equate cross-platform development with Windows 95 and NT on Intel platforms, you'll have no problems with OT Framework. Its VB-centric approach restricts you to those platforms.

In the final analysis, ComUnity OT Framework caters to very specific needs. If you're a VB developer building large-scale business applications and are willing to learn a new application framework, give version 1.0 of ComUnity OT Framework a closer look. Everyone else shoul d take a pass on it for now.

Product Information

ComUnity OT Framework.....$1000 per seat

(quantity discount available)

Siemens Nixdorf
Information Systems, Inc.
Burlington, MA
Phone: (800) 225-1484
Circle 983 on Inquiry Card.
Siemens Nixdorf Informationssysteme AG
Munich, Germany
Circle 984 on Inquiry Card.

HotBYTEs
- information on products covered or advertised in BYTE

Ratings

```
Technology      ****
Implementation  ***

Key

***** Outstanding
 **** Very Good
  *** Good
   ** Fair
    * Poor
```

Three Tiers for ComUnity

illustration_link (26 Kbytes) 

Implementing ComUnity applications as OLE servers lets them communicate over a network.

Tailor Your UI with ComUnity

screen_link (30 Kbytes) 

ComUnity OT Framework's Tailoring Tool lets users modify a custom application's UI.

Volker Weber and Hans-Jochen Schmitt are principals at ifos, a consulting firm in Darmstadt, Germany. They can be reached at 100120 .577@compuserve.com .